समरक्ष

Ubiquitous Actualization

eMote User's Manual for the .NOW

The Samraksh Company
September 2015

http://www.samraksh.com/

Contents

1. Origins of the eMote .NOW Platform

The wireless sensing network community was among Samraksh's earliest customers. This group had specific needs, such as reliable wireless networks that would work when sensors were deployed with a variety of topological and power management configurations.

Samraksh developed robust built-in wireless and low power management solutions. Samraksh also noted that not every wireless sensing customer wanted to use the same set of sensors. To Samraksh, this meant an opportunity to develop a more general solution in which sensors could be integrated with the basic platform.

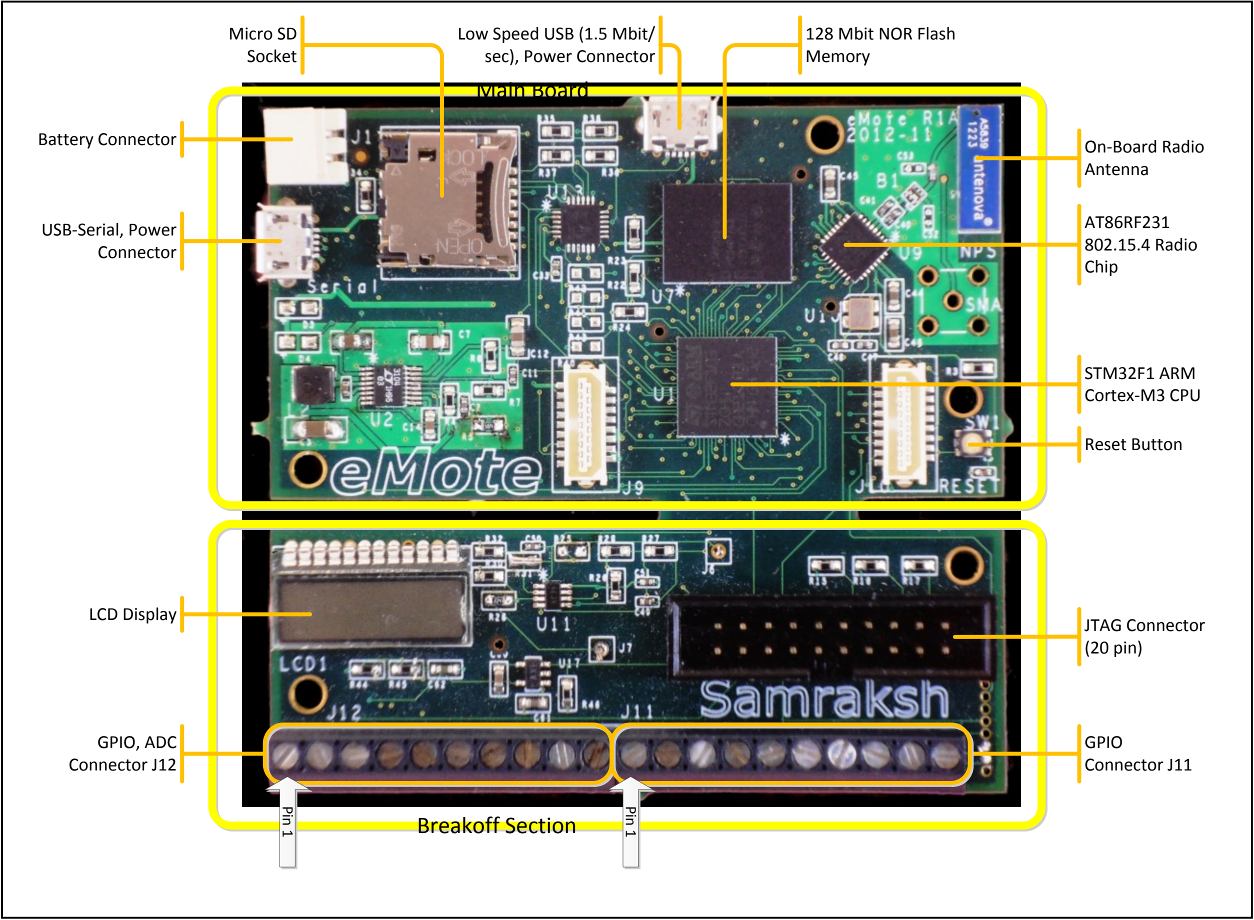
It's been a number of years now and the demand for wireless networking is only growing. Meanwhile, the maker / hobbyist community has also been growing. This community has a number of options for microcontrollers that support, well, various projects that we want to do. Recent entries include the Arduino and the Netduino.

The eMote .NOW is geared to supporting the maker/hobbyist community. While it delivers wireless networking out of the box, it also offers other features.

Terminology. ".NOW" refers to the physical board. "eMote" refers to the pre-installed software that lets you write programs in C#.

2. Layout of the .NOW Platform

The figures below show the major components of the .NOW. For details of the pinouts, see https://samraksh.com/support/layouts-pinouts-interconnects/dot-now.



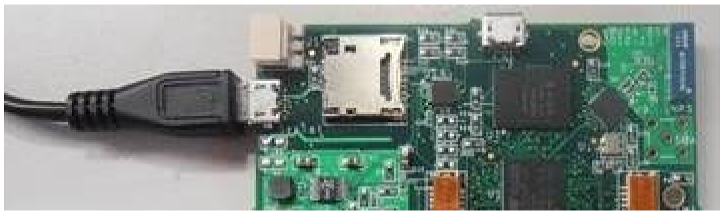
Note the on-board radio antenna, the two high density connectors, the microSD socket, the power connectors and the CPU on the main board area. The attached breakoff area has the LCD, the JTAG and the low density connectors. If you need a smaller form factor, the breakoff area can be physically removed by cutting through the connection part of the board.

3. Getting Started

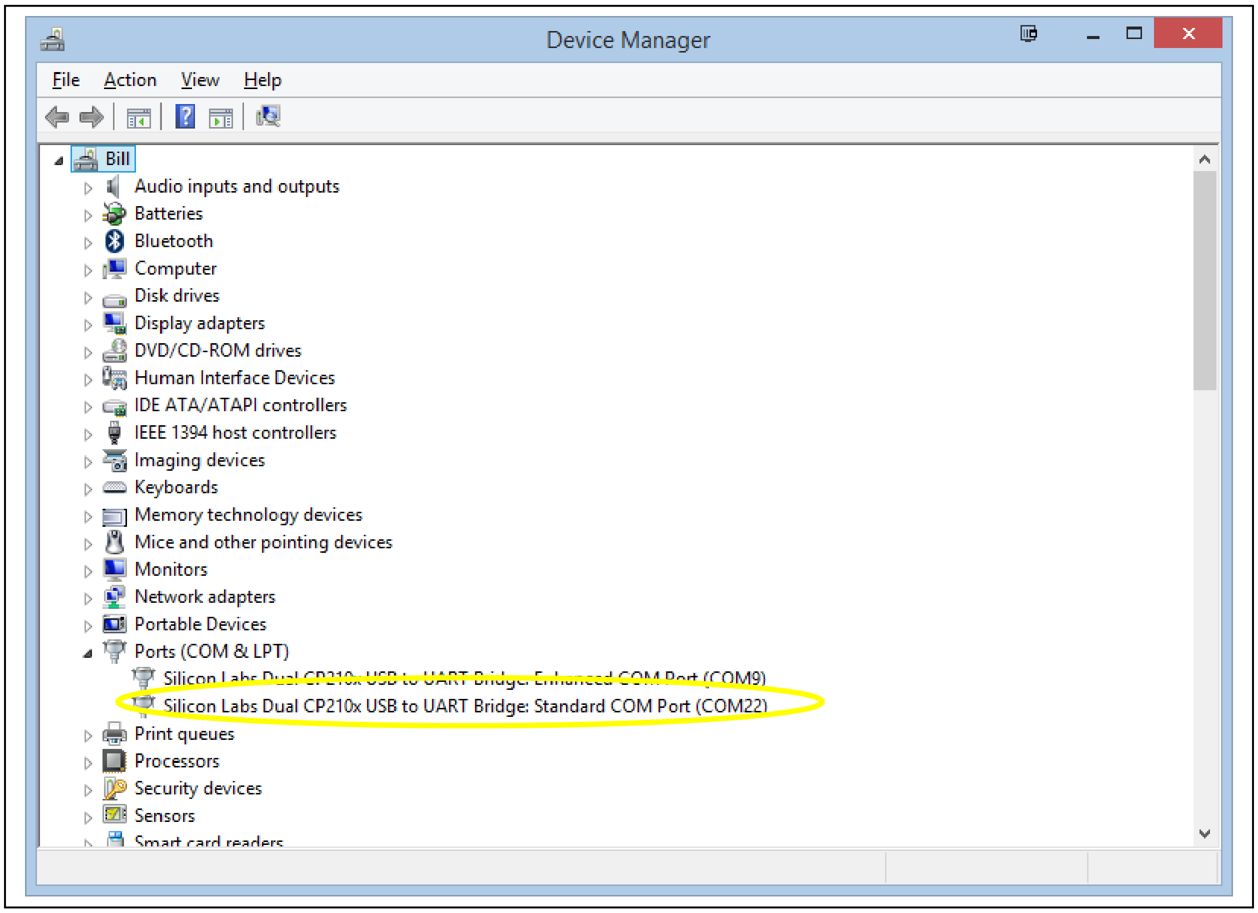
1. **Install Visual Studio 2012.** Visual Studio 2012 Windows Desktop is used as a C# development environment for the eMote. If you don't have it installed, you can get the free Express version from http://www.microsoft.com/visualstudio/eng/downloads. (The paid version might be offered first; scroll down to find the Visual Studio 2012 Express for Windows Desktop version.) Make sure you get the "Desktop" version as there are two Visual Studio Express 2012 products.
2. **Install Micro Framework SDK.** Get SDK version 4.3 from http://netmf.codeplex.com/releases/view/81000.

3. **Download eMote.** eMote consists of firmware that's flashed onto your .NOW and libraries that you can include for doing such things displaying on the LCD. You can get eMote from http://samraksh.com/support/emote-now-downloads. Your .NOW should already be flashed with the latest version of the eMote firmware. If not, see Section 6: Appendix: Updating the eMote Firmware.

4. **Install the Serial over USB driver.** Connect a standard Micro-B USB cable to the connector (located on the upper left of the .NOW board) to your PC. See the picture below. Typically, the operating system should be able to download the necessary driver and install it. If there are problems with the driver installation, you can get it from http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx.



5. **Confirm that the .NOW is connected.** Open Device Manager (Windows Start → Device Manager) and expand Ports (COM & LPT). In the screenshot, you see two COM ports, Enhanced on COM9 and standard on COM22. We'll be using the Standard port



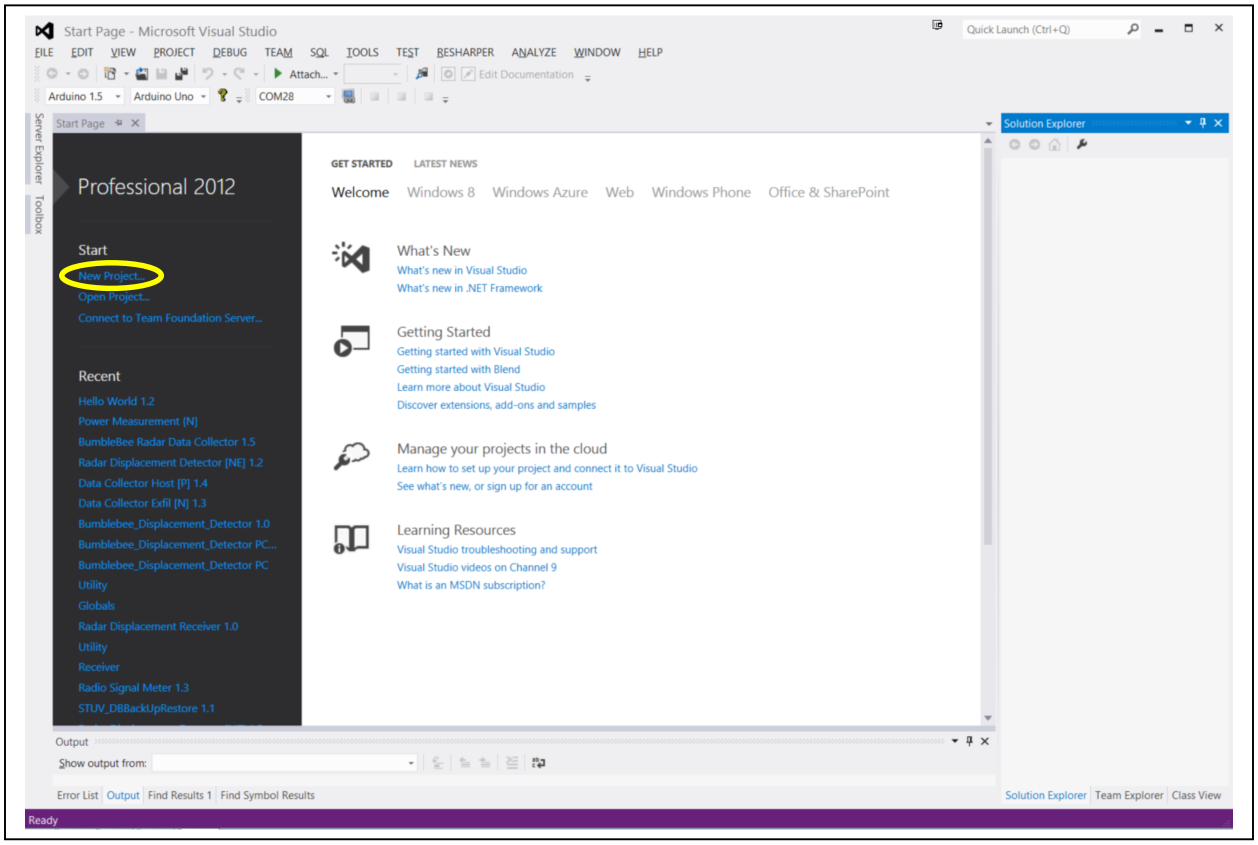
Bill
- Audio inputs and outputs
- Batteries
- Bluetooth
- Computer
- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Human Interface Devices
- IDE ATA/ATAPI controllers
- IEEE 1394 host controllers
- Imaging devices
- Keyboards
- Memory technology devices
- Mice and other pointing devices
- Monitors
- Network adapters
- Portable Devices
- Ports (COM & LPT)
  - Silicon Labs Dual CP210x USB to UART Bridge: Enhanced COM Port (COM9)
  - Silicon Labs Dual CP210x USB to UART Bridge: Standard COM Port (COM22)
- Print queues
- Processors
- Security devices
- Sensors
- Smart card readers

4. Your First Program

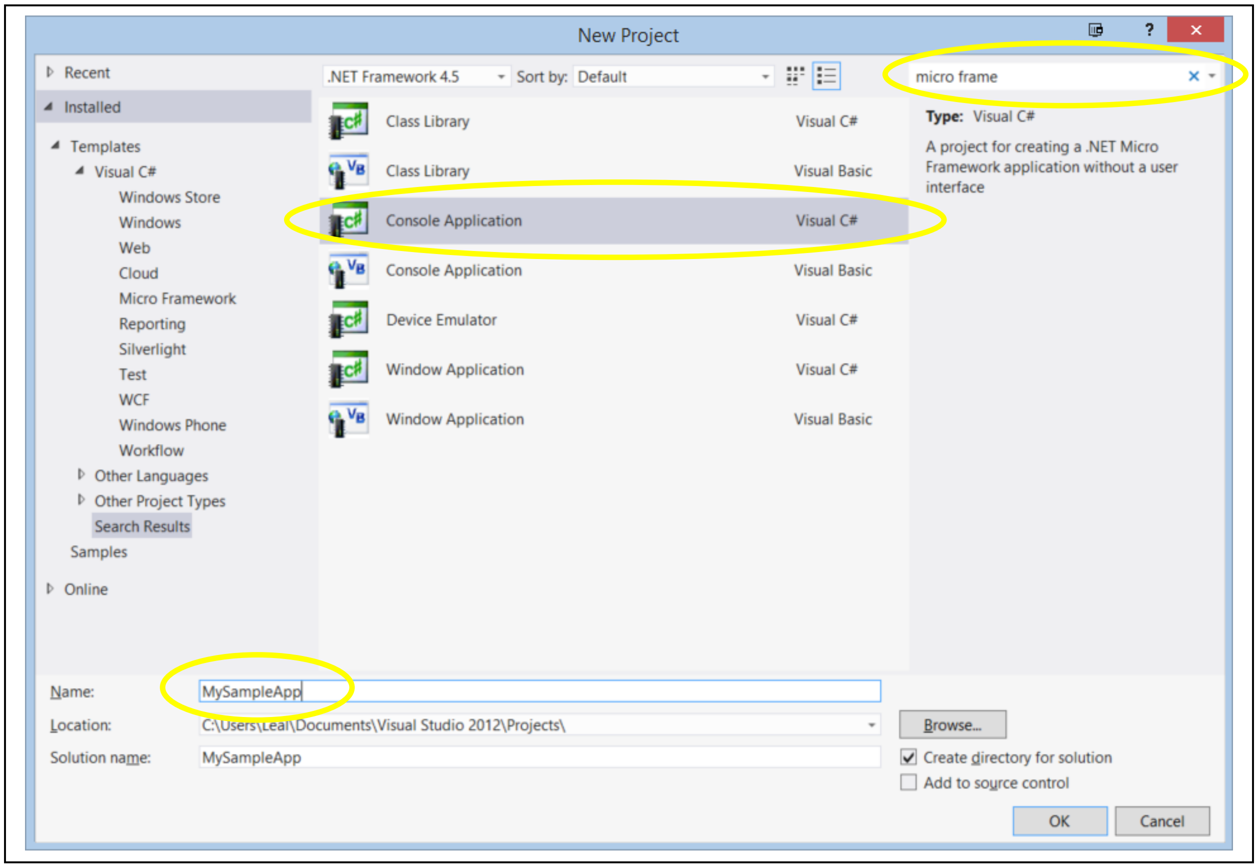
We'll go step by step on how to create and run an eMote .NOW program.

4.1 Using Visual Studio to Create and Deploy a Program

1. Open Visual Studio 2012. You'll see a window that looks something like this:


Start Page - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT DEBUG TEAM SQL TOOLS TEST RESHARPER ANALYZE WINDOW HELP
Attach...
Edit Documentation
Arduino 1.5
Arduino Uno
COM28
Start Page
Server Explorer
Toolbox
Professional 2012
Start
New Project...
Open Project...
Connect to Team Foundation Server...
Recent
Hello World 1.2
Power Measurement (N)
BumbleBee Radar Data Collector 1.5
Radar Displacement Detector (NE) 1.2
Data Collector Host (P) 1.4
Data Collector Exfil (N) 1.3
Bumblebee_Displacement_Detector 1.0
Bumblebee_Displacement_Detector PC...
Bumblebee_Displacement_Detector PC
Utility
Globals
Radar Displacement Receiver 1.0
Utility
Receiver
Radio Signal Meter 1.3
STUV_DBBackUpRestore 1.1
GET STARTED
LATEST NEWS
Welcome
Windows 8
Windows Azure
Web
Windows Phone
Office & SharePoint
What's New
What's new in Visual Studio
What's new in .NET Framework
Getting Started
Getting started with Visual Studio
Getting started with Blend
Learn more about Visual Studio
Discover extensions, add-ons and samples
Manage your projects in the cloud
Learn how to set up your project and connect it to Visual Studio
See what's new, or sign up for an account
Learning Resources
Visual Studio troubleshooting and support
Visual Studio videos on Channel 9
What is an MSDN subscription?
Solution Explorer
Output
Show output from:
Error List
Output
Find Results 1
Find Symbol Results
Solution Explorer
Team Explorer
Class View
Ready

2. Click on Start in the left hand pane. You'll see a window like the following. In the upper right, type in "micro frame" to look for Micro Framework project types. Choose a Console Application for Visual C#. Name it MySampleApp and put it in the VS 2012 Projects folder.



micro frame

Class Library	Visual C#
Class Library	Visual Basic
Console Application	Visual C#
Console Application	Visual Basic
Device Emulator	Visual C#
Window Application	Visual C#
Window Application	Visual Basic

Type: Visual C#

A project for creating a .NET Micro Framework application without a user interface

Name: MySampleApp

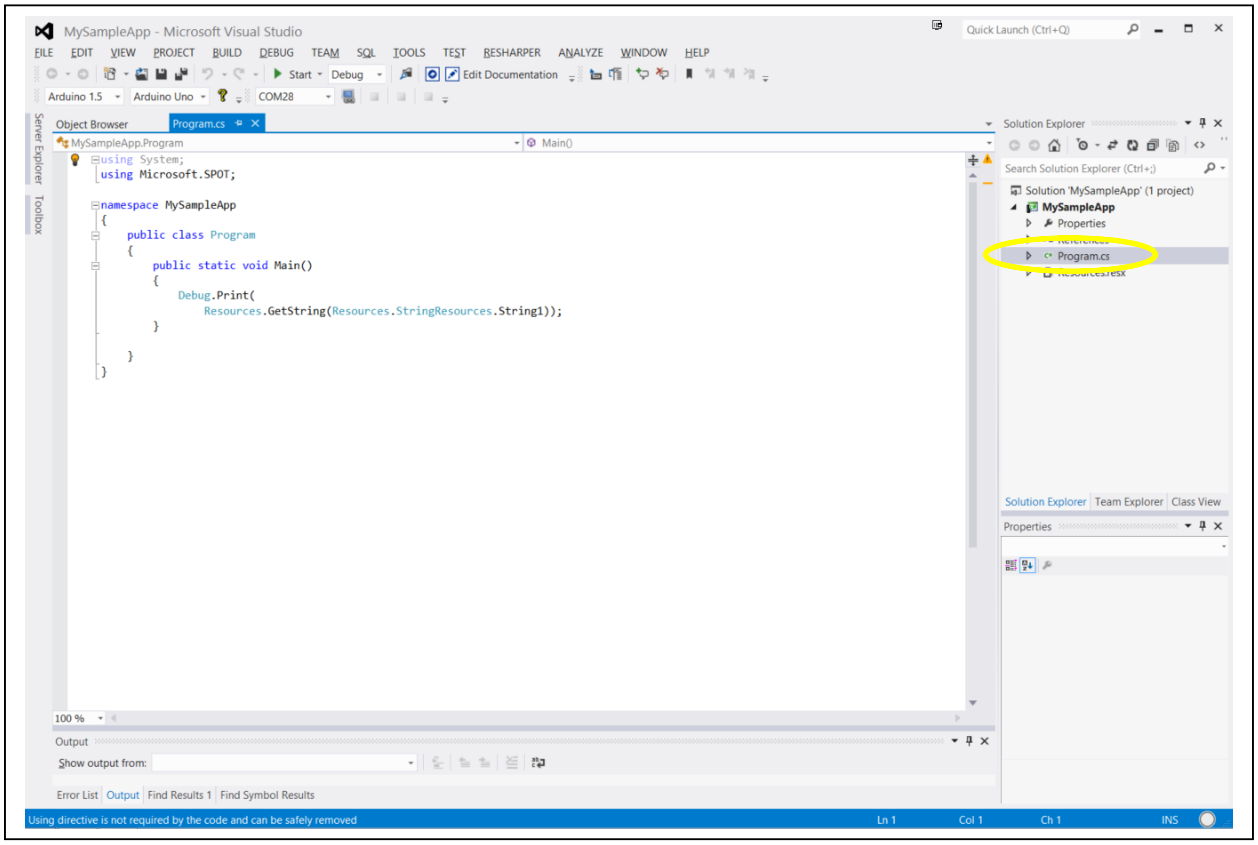
Location: C:\Users\Leal\Documents\Visual Studio 2012\Projects\

Solution name: MySampleApp

[✓] Create directory for solution

[ ] Add to source control

3. Click OK. When the solution is created, open Program.cs:

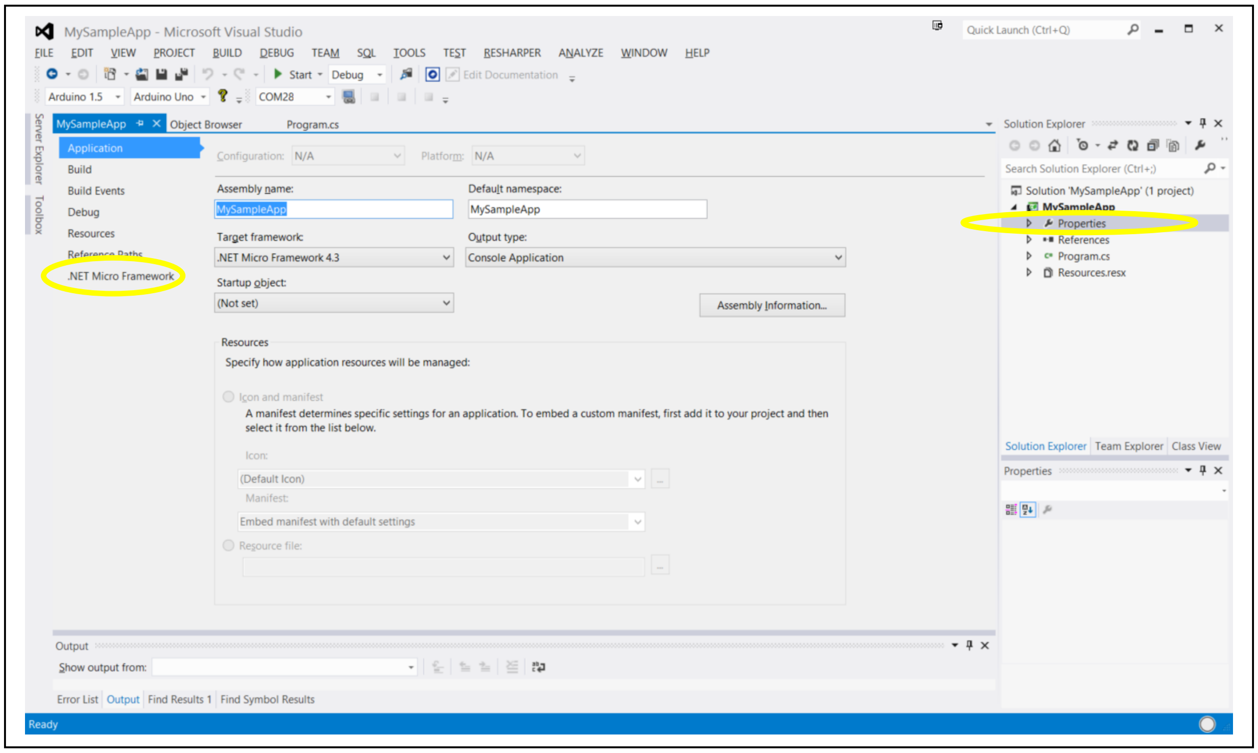


```
using System;
using Microsoft.SPOT;

namespace MySampleApp
{
    public class Program
    {
        public static void Main()
        {
            Debug.Print(
                Resources.GetString(Resources.StringResources.String1));
        }

    }
}
```

Using directive is not required by the code and can be safely removed

This program will print the name of the program in the Debug window. To run this program connect a .NOW via USB using the instructions given above in Section 3: Getting Started. Make note of the number of the Standard COM port. For illustration, we'll assume it's COM22.

4. Double-click on Properties in the right hand panel. The MySampleApp Properties window will open.

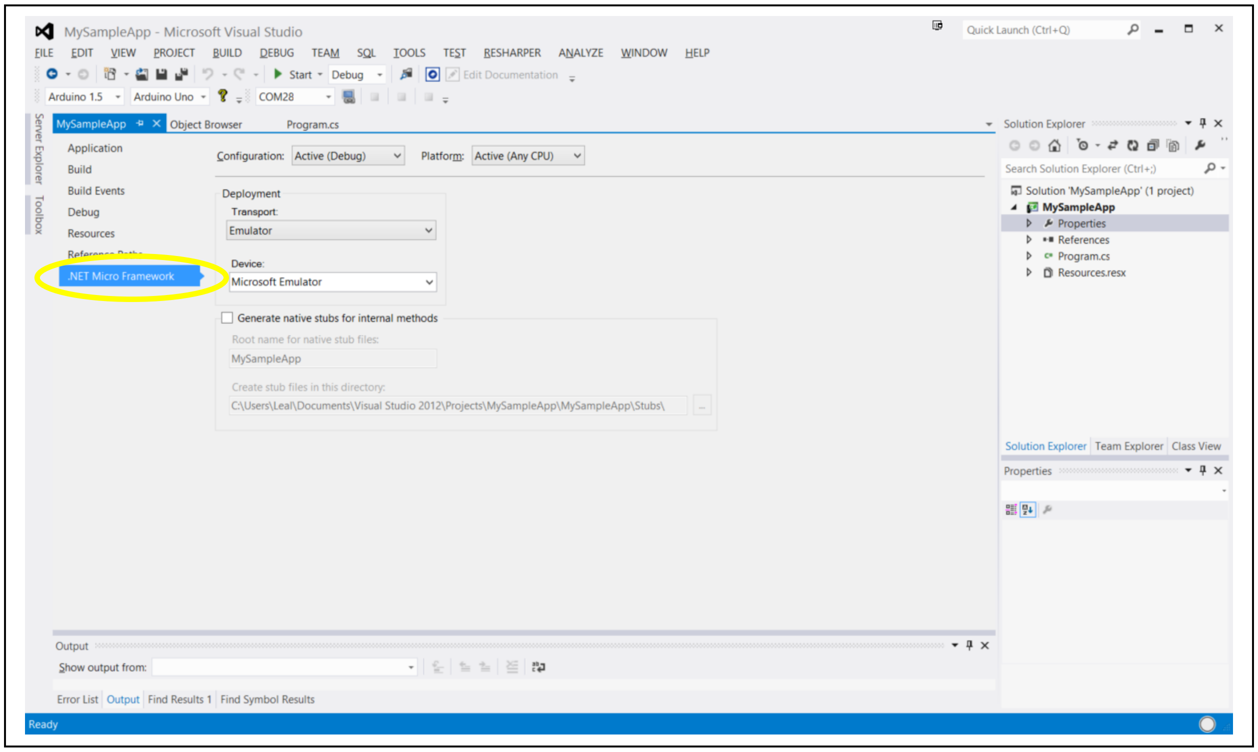

MySampleApp - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST RESHARPER ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Start Debug Edit Documentation
Arduino 1.5 Arduino Uno COM28
MySampleApp Object Browser Program.cs
Application
Build
Build Events
Debug
Resources
Reference Paths
.NET Micro Framework
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MySampleApp' (1 project)
MySampleApp
Properties
References
Program.cs
Resources.resx
Solution Explorer Team Explorer Class View
Properties
Output
Show output from:
Error List Output Find Results 1 Find Symbol Results
Ready

Configuration: N/A Platform: N/A

Assembly name: MySampleApp
Default namespace: MySampleApp
Target framework: .NET Micro Framework 4.3
Output type: Console Application
Startup object: (Not set)
Assembly Information...

Resources
Specify how application resources will be managed:
Icon and manifest
A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.
Icon: (Default Icon)
Manifest: Embed manifest with default settings
Resource file:

5. Then click on .Net Micro Framework.

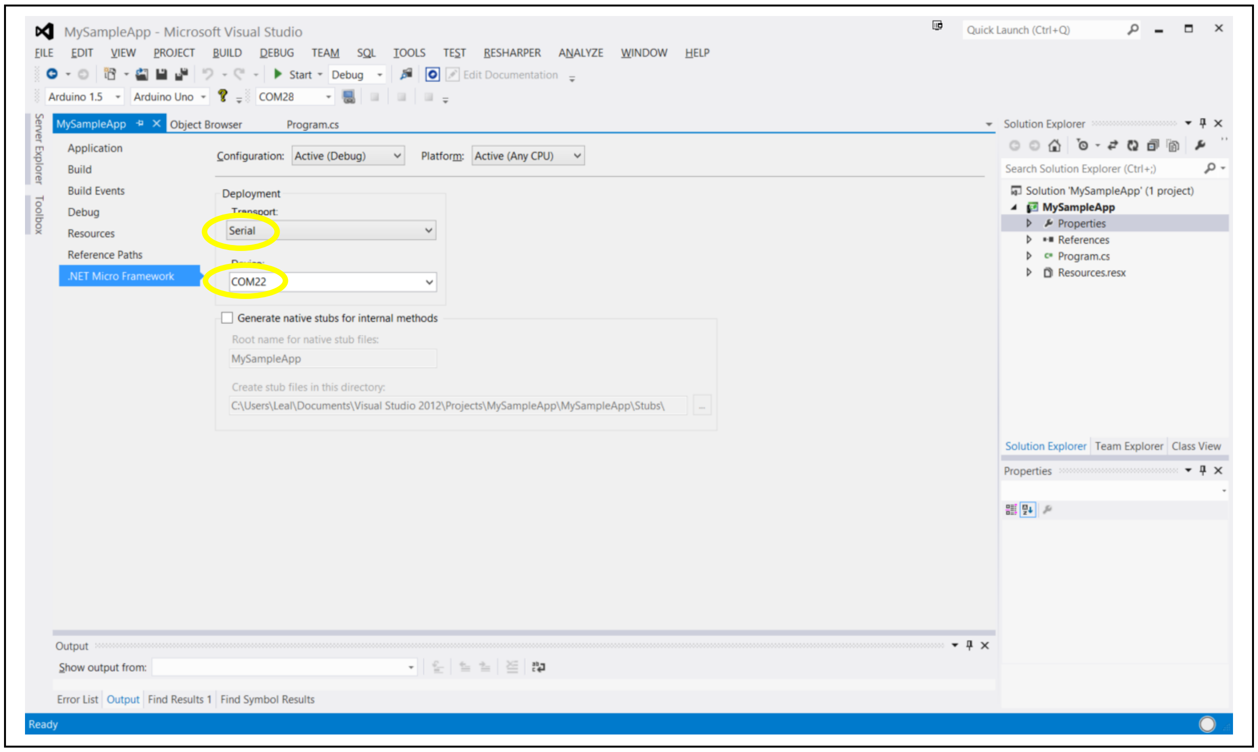

MySampleApp - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST RESHARPER ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Start Debug Edit Documentation
Arduino 1.5 Arduino Uno COM28
MySampleApp Object Browser Program.cs
Application
Build
Build Events
Debug
Resources
Reference Paths
.NET Micro Framework
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MySampleApp' (1 project)
MySampleApp
Properties
References
Program.cs
Resources.resx
Solution Explorer Team Explorer Class View
Properties
Output
Show output from:
Error List Output Find Results 1 Find Symbol Results
Ready

Configuration: Active (Debug) Platform: Active (Any CPU)

Deployment
Transport: Emulator
Device: Microsoft Emulator

Generate native stubs for internal methods
Root name for native stub files: MySampleApp
Create stub files in this directory: C:\Users\Leal\Documents\Visual Studio 2012\Projects\MySampleApp\MySampleApp\Stubs\

6. In the Deployment section, click on Transport and chose Serial. Then click on Device and chose the Standard COM port; COM22 in this case.

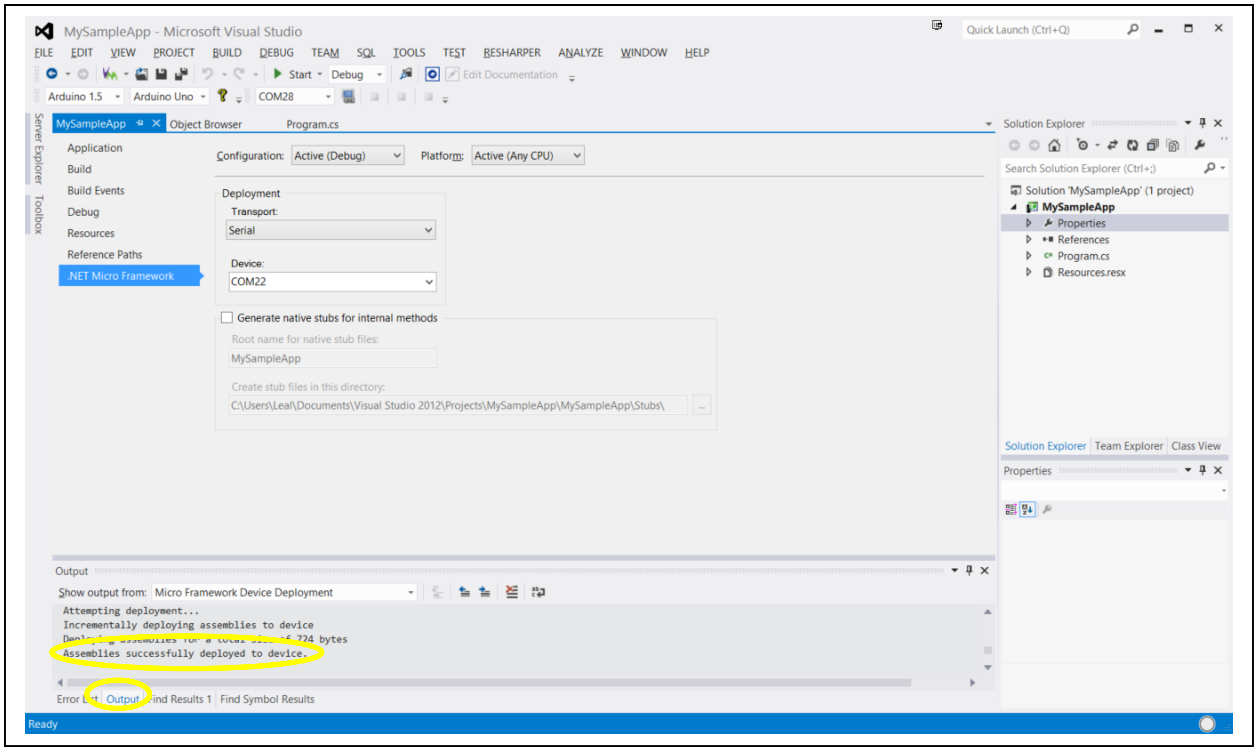

MySampleApp - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST RESHARPER ANALYZE WINDOW HELP
Start Debug
Arduino 1.5 Arduino Uno COM28
MySampleApp Object Browser Program.cs
Application
Build
Build Events
Debug
Resources
Reference Paths
.NET Micro Framework
Solution Explorer
Solution 'MySampleApp' (1 project)
MySampleApp
Properties
References
Program.cs
Resources.resx
Solution Explorer Team Explorer Class View
Properties
Output
Show output from:
Error List Output Find Results 1 Find Symbol Results
Ready

Configuration: Active (Debug)
Platform: Active (Any CPU)

Deployment
Transport: Serial
Device: COM22

Generate native stubs for internal methods
Root name for native stub files: MySampleApp
Create stub files in this directory: C:\Users\Leal\Documents\Visual Studio 2012\Projects\MySampleApp\MySampleApp\Stubs\

7. Now we're ready to deploy the application to the .NOW; that is, to compile the program and copy the results to the .NOW for execution. In the menu, click Build → Deploy Solution. You'll see messages appearing in the Output window. If all goes well, it will say "Assemblies successfully deployed to device".



Configuration: Active (Debug)
Platform: Active (Any CPU)

Deployment
Transport: Serial
Device: COM22

Generate native stubs for internal methods
Root name for native stub files: MySampleApp
Create stub files in this directory: C:\Users\Leal\Documents\Visual Studio 2012\Projects\MySampleApp\MySampleApp\Stubs\

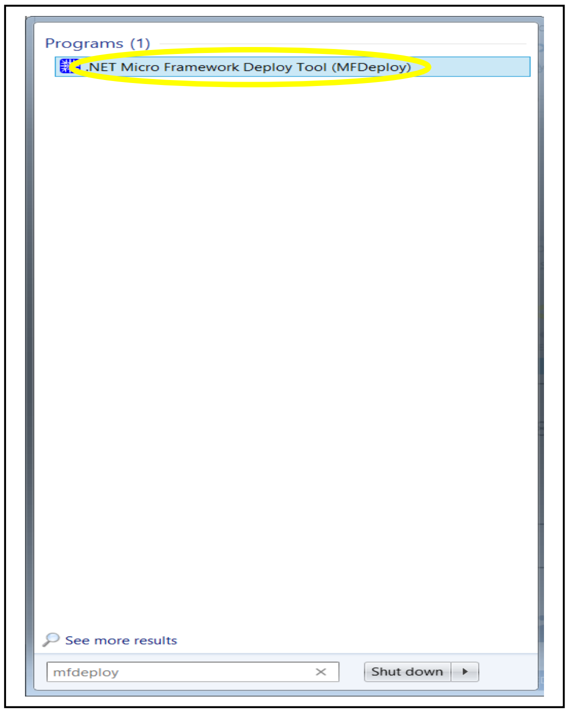
Show output from: Micro Framework Device Deployment

```
Attempting deployment...
Incrementally deploying assemblies to device
Deploying assemblies for a total size of 724 bytes
Assemblies successfully deployed to device.
```

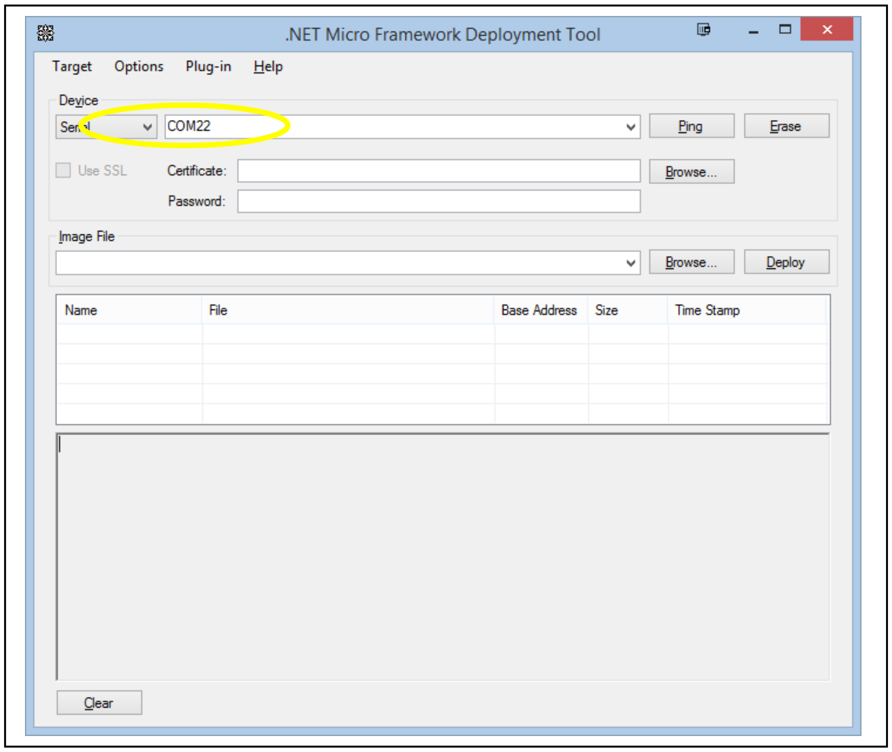
4.2 Using MFDeploy to Run and Test

So now we're ready to actually run the program. There are two ways to do this. First, we'll use MFDeploy, an application that comes with the Micro Framework SDK you installed earlier. MFDeploy, among other things, lets you see the Debug messages that the eMote .NOW program generates.

1. In the Windows Start search box, type in "MFDeploy".

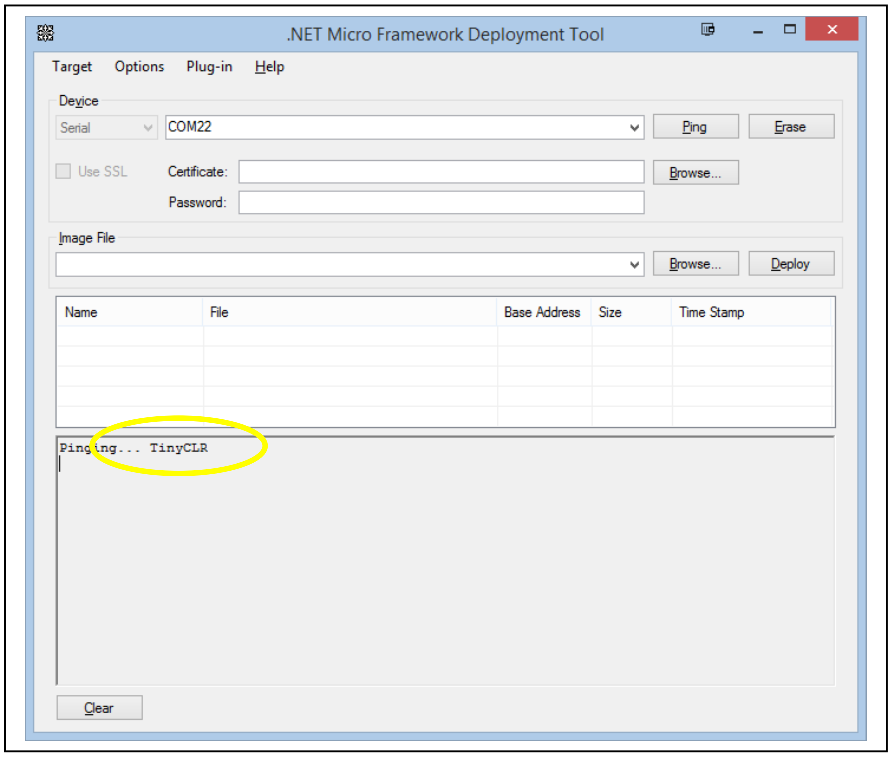


2. Click on the program it finds. When it opens, make sure the Device is Serial and that COM22 is selected.



Device: Serial
COM22

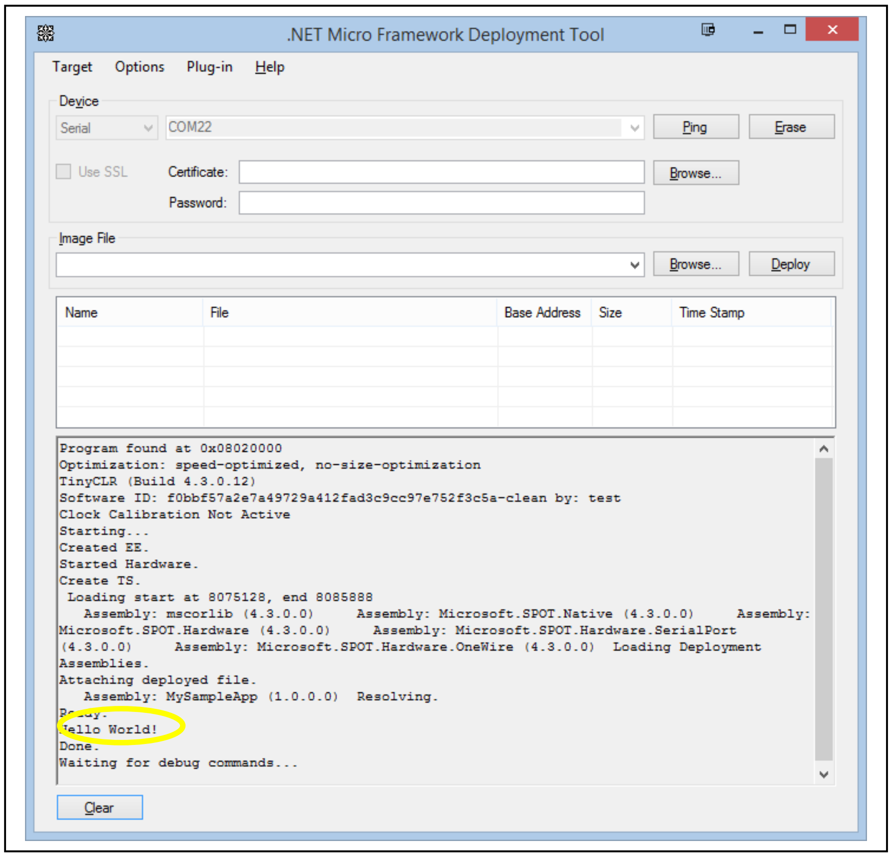
3. Next click on Ping. You should see TinyCLR reported:



Device: Serial
COM22

```
Pinging... TinyCLR
```

4. In MFDeploy, click on Target → Connect in the menu (or press F5). Then restart the .NOW by pressing the Reset button located on the right hand side of the .NOW. See the annotated photo in Section 2: Layout of the .NOW Platform, above. You should see the initialization information followed by the program output, “Hello World!”.



Device: Serial COM22

Use SSL

Certificate:

Password:

Image File

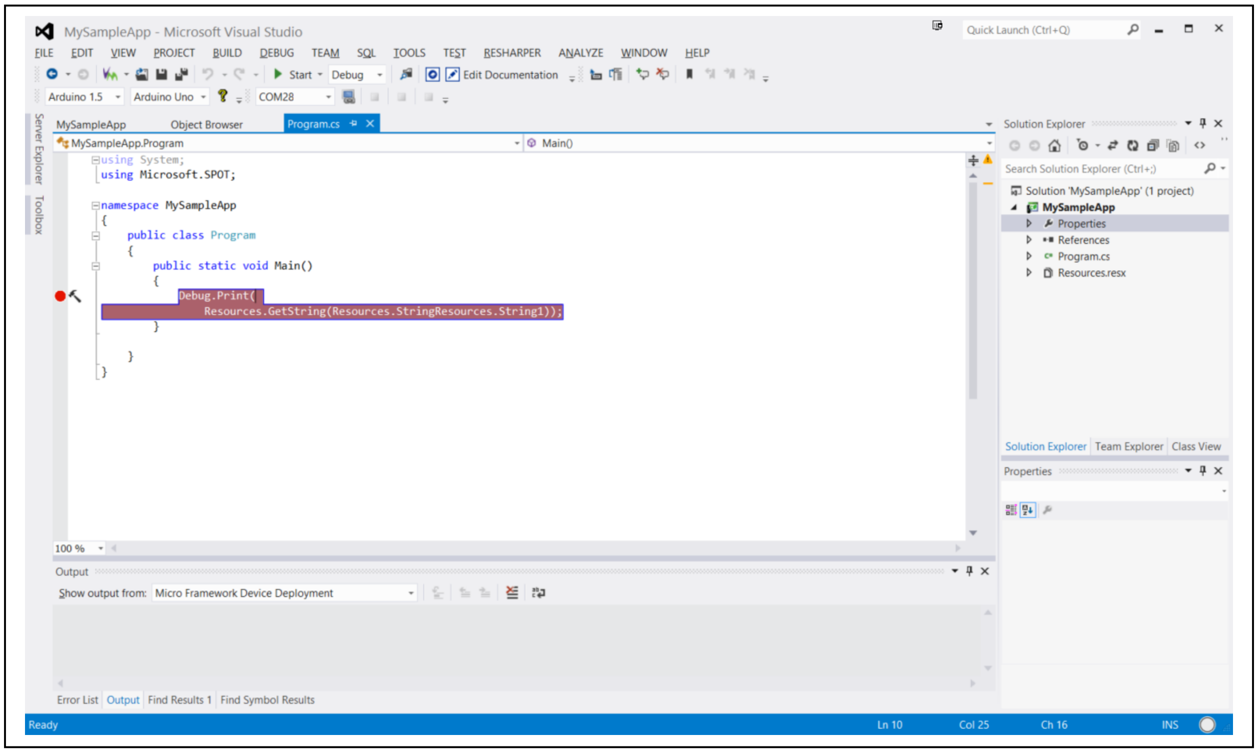
```
Program found at 0x08020000
Optimization: speed-optimized, no-size-optimization
TinyCLR (Build 4.3.0.12)
Software ID: f0bbf57a2e7a49729a412fad3c9cc97e752f3c5a-clean by: test
Clock Calibration Not Active
Starting...
Created EE.
Started Hardware.
Create TS.
 Loading start at 8075128, end 8085888
   Assembly: mscorlib (4.3.0.0)     Assembly: Microsoft.SPOT.Native (4.3.0.0)     Assembly:
Microsoft.SPOT.Hardware (4.3.0.0)     Assembly: Microsoft.SPOT.Hardware.SerialPort
(4.3.0.0)     Assembly: Microsoft.SPOT.Hardware.OneWire (4.3.0.0)  Loading Deployment
Assemblies.
Attaching deployed file.
   Assembly: MySampleApp (1.0.0.0)  Resolving.
Ready.
Hello World!
Done.
Waiting for debug commands...
```

5. When you’re done, Target → Disconnect in the menu (or press ctrl-F5).

4.3 Using Visual Studio for Interactive Testing and Debugging

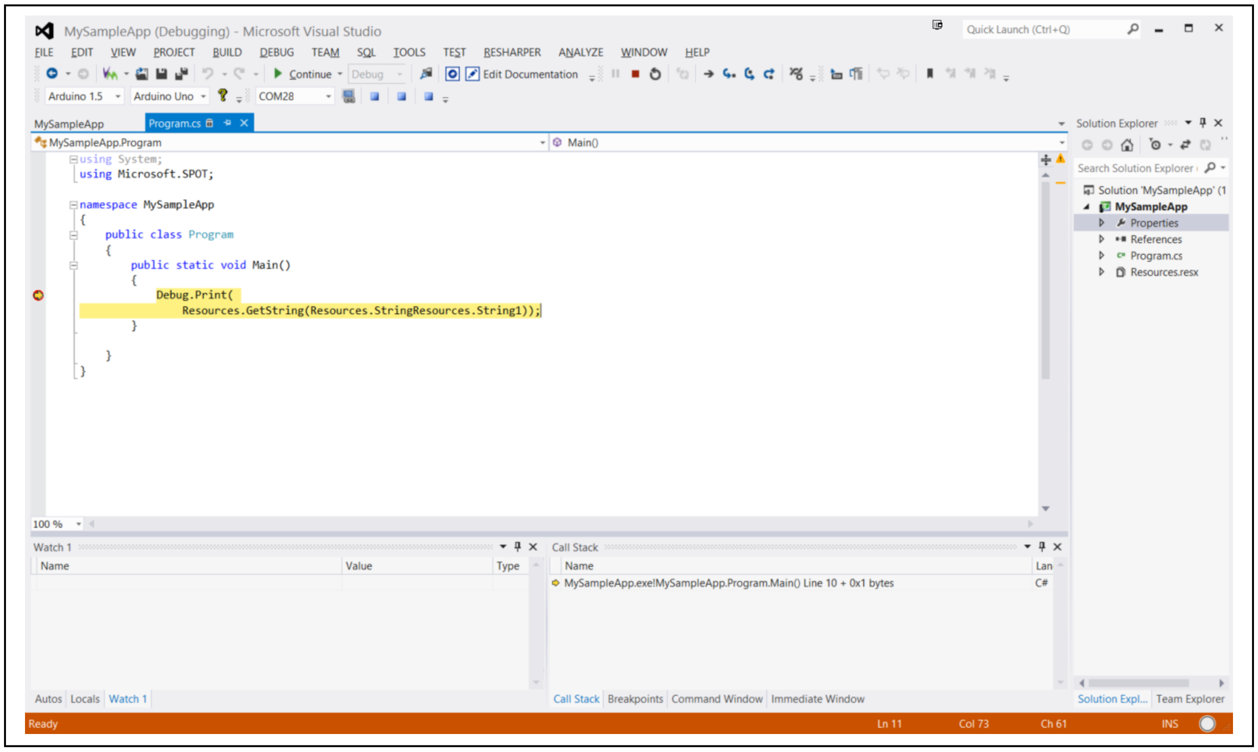
Congratulations. You've just completed and tested your first eMote .NOW program. But there's another way to test the program. eMote on the .NOW also supports interactive, on-board debugging. This is a powerful feature that you can use to track down bugs in your program.

1. Go back to Visual Studio and select the Program window. On the Debug.Print line, press F9 to set a breakpoint.



```
using System;
using Microsoft.SPOT;

namespace MySampleApp
{
    public class Program
    {
        public static void Main()
        {
            Debug.Print(
                Resources.GetString(Resources.StringResources.String1));
        }

    }
}
```

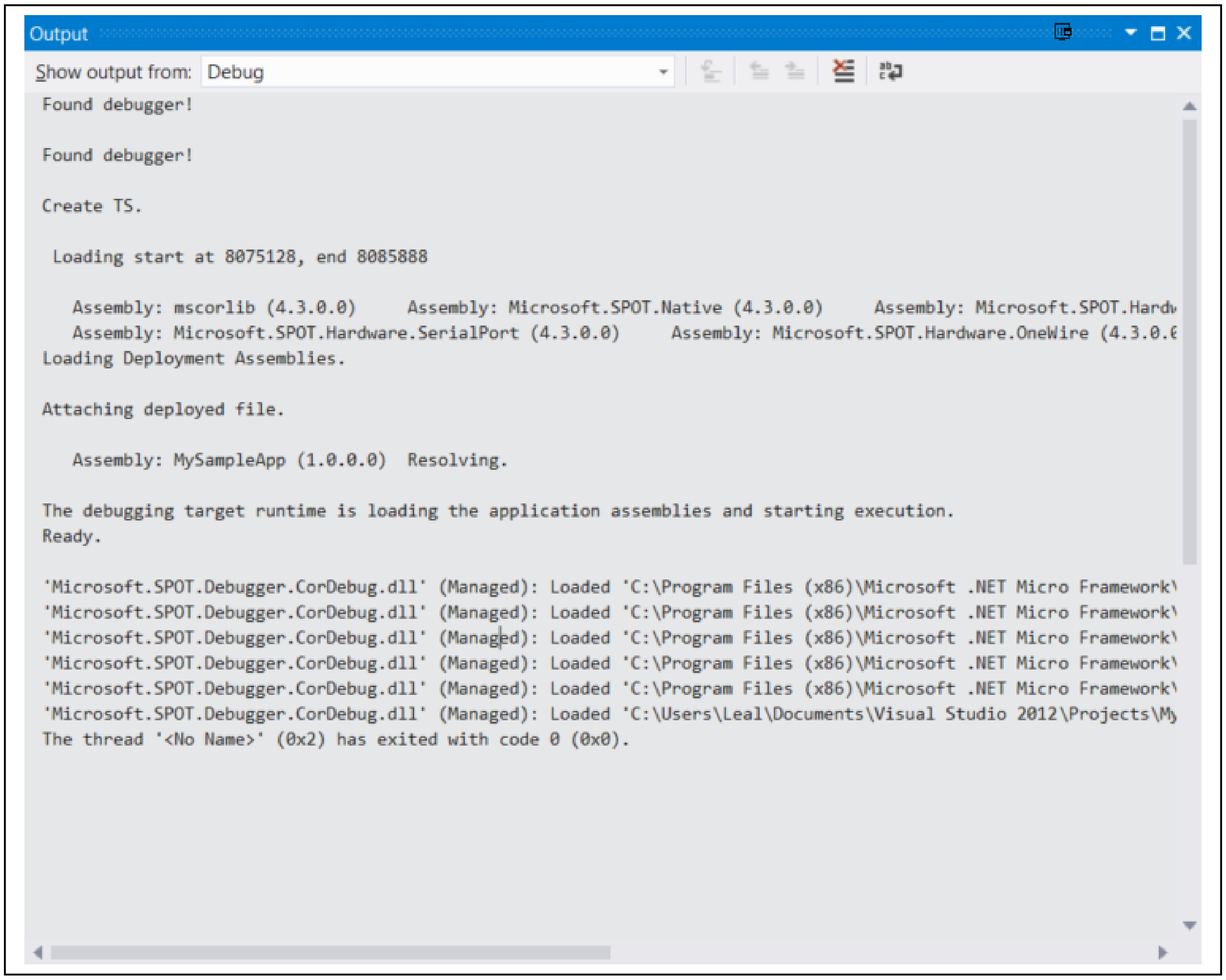
2. Next click on Debug → Start Debugging or press F5. You'll see some messages in the Debug window. After a moment, your Debug window might disappear; if so, you can click on View → Output to see it. If all goes well, you'll see the breakpoint hit.



```
using System;
using Microsoft.SPOT;

namespace MySampleApp
{
    public class Program
    {
        public static void Main()
        {
            Debug.Print(
                Resources.GetString(Resources.StringResources.String1));
        }

    }
}
```

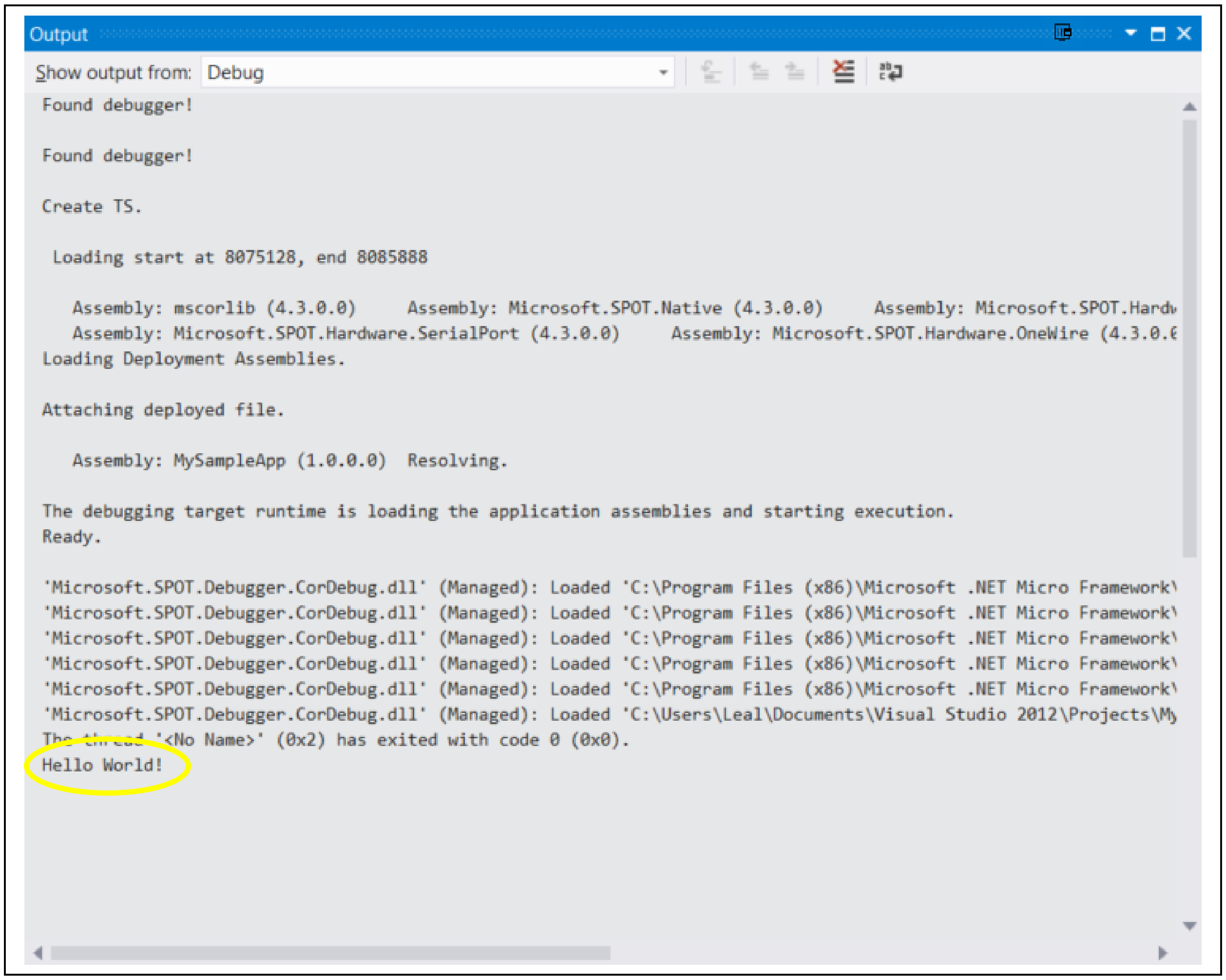
```
MySampleApp.exe!MySampleApp.Program.Main() Line 10 + 0x1 bytes    C#
```

3. In the Output window, you'll see pretty much the same kinds of messages as with MFDeploy. You'll notice that at this point, the result of the Debug.Print hasn't been displayed. That's because it's hit the breakpoint and is waiting for you to do something.



```
Found debugger!

Found debugger!

Create TS.

 Loading start at 8075128, end 8085888

   Assembly: mscorlib (4.3.0.0)     Assembly: Microsoft.SPOT.Native (4.3.0.0)     Assembly: Microsoft.SPOT.Hardw
   Assembly: Microsoft.SPOT.Hardware.SerialPort (4.3.0.0)     Assembly: Microsoft.SPOT.Hardware.OneWire (4.3.0.6
Loading Deployment Assemblies.

Attaching deployed file.

   Assembly: MySampleApp (1.0.0.0)  Resolving.

The debugging target runtime is loading the application assemblies and starting execution.
Ready.

'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\Microsoft .NET Micro Framework\
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\Microsoft .NET Micro Framework\
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\Microsoft .NET Micro Framework\
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\Microsoft .NET Micro Framework\
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\Microsoft .NET Micro Framework\
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Users\Leal\Documents\Visual Studio 2012\Projects\My
The thread '<No Name>' (0x2) has exited with code 0 (0x0).
```

4. At this point you can go ahead and press F10 to execute the next instruction. You'll see the break-point move forward and the result in the Output window.



```
Found debugger!

Found debugger!

Create TS.

 Loading start at 8075128, end 8085888

   Assembly: mscorlib (4.3.0.0)     Assembly: Microsoft.SPOT.Native (4.3.0.0)     Assembly: Microsoft.SPOT.Hardw
   Assembly: Microsoft.SPOT.Hardware.SerialPort (4.3.0.0)     Assembly: Microsoft.SPOT.Hardware.OneWire (4.3.0.
Loading Deployment Assemblies.

Attaching deployed file.

   Assembly: MySampleApp (1.0.0.0)  Resolving.

The debugging target runtime is loading the application assemblies and starting execution.
Ready.

'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\Microsoft .NET Micro Framework\
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\Microsoft .NET Micro Framework\
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\Microsoft .NET Micro Framework\
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\Microsoft .NET Micro Framework\
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\Microsoft .NET Micro Framework\
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Users\Leal\Documents\Visual Studio 2012\Projects\My
The thread '<No Name>' (0x2) has exited with code 0 (0x0).
Hello World!
```

For more about how to use the debugger, see for example http://www.dotnetperls.com/debugging.

4.4 MFDeploy vs. Visual Studio for Testing and Debugging

Now you have two ways to do testing and debugging. Which is best? As usual, it depends …

Visual Studio is good if you need to step through some code to check what's happening and to check the values of the variables it's using. However, the program runs considerably slower and this could be a problem for time-critical code. Also, since Visual Studio has to connect to the debugger on the eMote .NOW, it takes longer to start up.

MFDeploy, by contrast, is light-weight, since it's merely interacting via the serial connection. However, you can't use breakpoints or do the other immediate things that the interactive debugging of Visual Studio enables.

5. Wireless and the eMote .NOW

5.1 Wireless Networking in the eMote .NOW

The .NOW offers built-in wireless capability. This wireless capability uses the IEEE 802.15.4 standard. eMote .NOW is not the only product making use of IEEE 802.15.4. In order to highlight some of the differences in capability offered by the eMote, it is useful to review the 802.15.4 standards. (IEEE is the Institute of Electrical and Electronics Engineers).

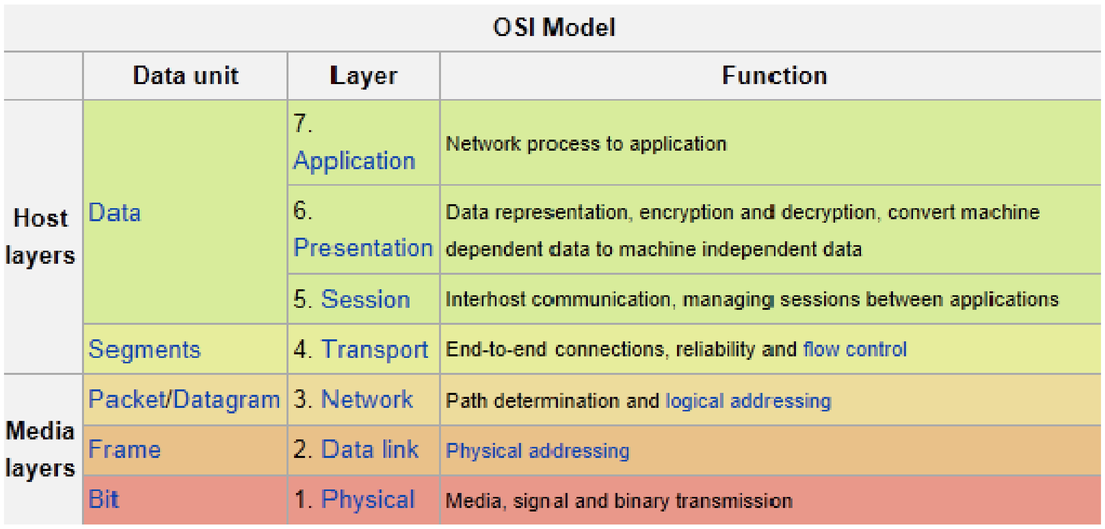
5.2 Background: Network and Network Stack

As part of this review, we consider the ‘network stack’ concept. The network stack is an abstraction that maps the functionality that needed to make networking a reality to different layers. Each layer offers a specific and related set of functionality and the layers are stacked on top of one another to provide the complete functionality for networking. Usually in any networking stack the lower layers provide simpler functions, which are in turn used by the higher layers to provide increasingly complex functionality and end-to-end semantics.

It is helpful to start with the conceptualization of a network stack for wired networking. The OSI (Open Systems Interconnection) standard model from the International Standards Organization (ISO 7498-1) is shown in the table below. In implementation and practice, the layers 5 through 7 are folded together. The functionality at each layer is governed by the standards. The standards-compliant implementation at each layer, which supports the required functionality, can be achieved by different variants which are typically termed protocols. For example, at the transport layer, TCP/IP protocols suite includes TCP and UDP.

In the case of wireless networking, IEEE 802.15.4 is one of the standards designed for low-power short-range networking. The focus of this standard is that of a wireless personal area network (WPAN). Using the OSI as a reference model, the IEEE 802.15.4 standard addresses the functionality of the two bottom layers, the physical layer and the MAC layer.

Key functionalities of the physical layer include managing the radio frequency of the sender/receiver (usually most radio standards operate in a number of frequencies, sometimes also referred to as radio channels), synchronizing the sending and receiving radios, and encoding/decoding the data transmitted



OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication, managing sessions between applications
	Segments	4. Transport	End-to-end connections, reliability and flow control
Media layers	Packet/Datagram	3. Network	Path determination and logical addressing
	Frame	2. Data link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

OSI Model (Adapted from Wikipedia. See http://en.wikipedia.org/wiki/OSI_model.

into physical layer symbols that are more suitable for transmission/reception. Key functionalities of the MAC layer include managing access to the physical channel and enabling MAC frame transmission. The MAC frame is the term given to the unit of data that will be transported. A MAC can have several types of frames, such as the data frame, the MAC command frame, an acknowledgement frame and a beacon frame. The beacon frame is used by an implementation that includes beaconing, which is a process that allows a node to announce its presence. A brief discussion of the details of the physical layer and of the MAC layer can be found at Wikipedia: http://en.wikipedia.org/wiki/IEEE_802.15.4.

A crucial point is that specifications for the upper layers of the network stack for wireless communication are not addressed by the 802.15.4 specs. This means that functionality that would be handled by the upper layers is open to different implementations and customized solutions.

5.3 MAC Implementation Approaches and Higher Level 'Solutions'

Product offerings may assume the needs of wireless applications and may provide their customized solution to implementations of the MAC as well as elements of upper layers. For example, in a wireless sensing application, the application infrastructure might include a way to organize the wireless nodes to allow them to pass their data from nodes to a collection point.

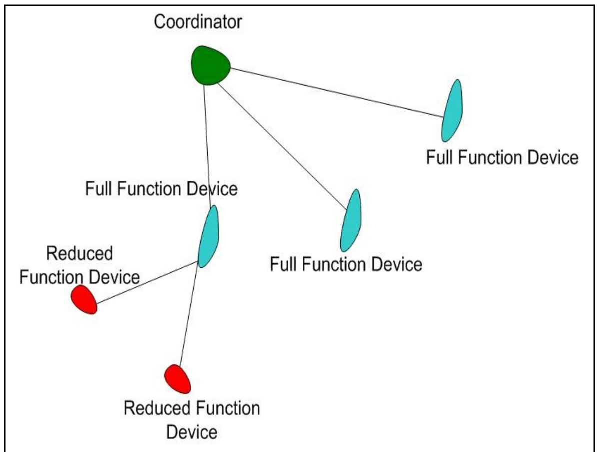
Zigbee: One of the integrated product offerings using the IEEE 802.15.4 standard for the physical and MAC layer is that of Zigbee solution. Zigbee is a networking standard for higher layers published by the Zigbee Alliance (a group of companies), that build on top of the 802.15.4 standard. This solution proposes distinct types of nodes: the full functioning nodes and those that are reduced function. The full functioning nodes can act as Coordinator nodes. The Coordinator concept plays a role in how the Zigbee solution sets up and maintains the network. The reduced function devices are only allowed to talk with one full function node.

This sets up a mesh network in which Coordinators talk to each other and each Coordinator or fully functioning device can have multiple reduced function devices which talk to it. Thus, passing a sensor reading from a reduced function device would involve a message to its associated fully functioning device. This device, if not a Coordinator, would be able to interact with a Coordinator and pass the message along the network as per the application-specific programming. For more information about Zigbee please see http://en.wikipedia.org/wiki/ZigBee.

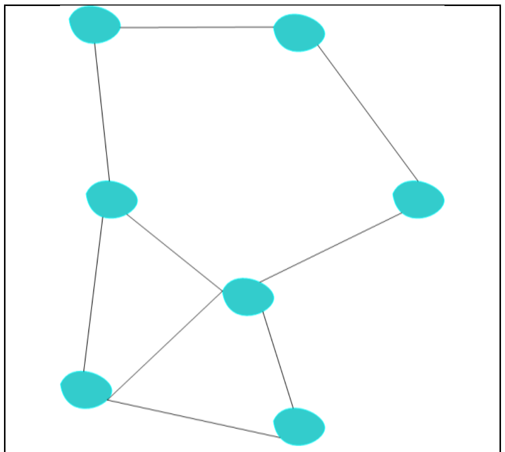
5.4 eMote Stack

Every node in the eMote network is a peer. The main difference between the eMote stack and the Zigbee stack is that the eMote stack considers the typical networking case to be peer-to-peer, whereas the Zigbee approaches uses Coordinator/full function/reduced function nodes. The eMote stack approach lets the user build whatever type of network structure they want on top of the peer-to-peer architecture. As one example, the peer-to-peer nodes could build routes through the network to a base station. As another example, the base station could host a GUI that allowed the user to process content from received messages and display resulting information.

The first figure below illustrates the Zigbee Coordinator-based network architecture; while the second illustrates eMote's peer-to-peer based architecture.



The coordinator network used by ZigBee.



The peer-to-peer network used by eMote.

There exists more than one type of MAC protocol that can be used. eMote at present uses a simple MAC, based on CSMA (carrier sense, multiple access). For more information, see http://en.wikipedia.org/wiki/Carrier_sense_multiple_access. An implementation of OMAC is planned. With most protocols, a receiving node has to be listening for network transmissions all the time and hence is usually the largest consumer of power. OMAC focuses on coordinating transmissions so that they are sent only when the receiver is listening. See http://dl.acm.org/citation.cfm?id=1318381 for more details

Access to the API for the MAC layer is provided by one of the Samraksh custom DLLs. There are additional MAC protocols being developed for the general eMote; selected candidates may be offered for the eMote .NOW platform in the future.

This table shows some of the .NOW network-related specifications.

Quantity	Value
Operating Frequency	2.4 GHz
Data Rate	250 kb/s
Antenna	SMA connector or built in chip antenna
Encryption	Not supported in software yet; AES module in hardware
Outdoor Range	100 m (Line of Sight, without any hindrance like foliage)
Indoor Range	Roughly 3-10 m (Through one wall)
Radio Chip	Atmel RF231
MAC Protocols	CSMA [for now, may be expanded].

5.5 Sample Wireless Application Notes

For examples of how to use the radio, see the following app notes:

- Radio Signal Meter https://samraksh.com/learning/app-notes/30-learning/app-notes/intro-app-notes/61-radio-signal-meter. Lets you determine the presence and quality of a radio signal for a

.NOW that's broadcasting on the same channel. The app note also sends out messages, so you can install it on two .NOWs and observe how each is hearing the other.

- Wireless Radio Ping-Pong (CSMA) https://samraksh.com/learning/app-notes/30-learning/app-notes/intro-app-notes/57-wireless-radio-ping-pong-csma. Motes converge to a common number that is incremented alternately by each mote after it receives a message from the other. In case one mote goes down or goes out of range, they will reconverge when both are available again.

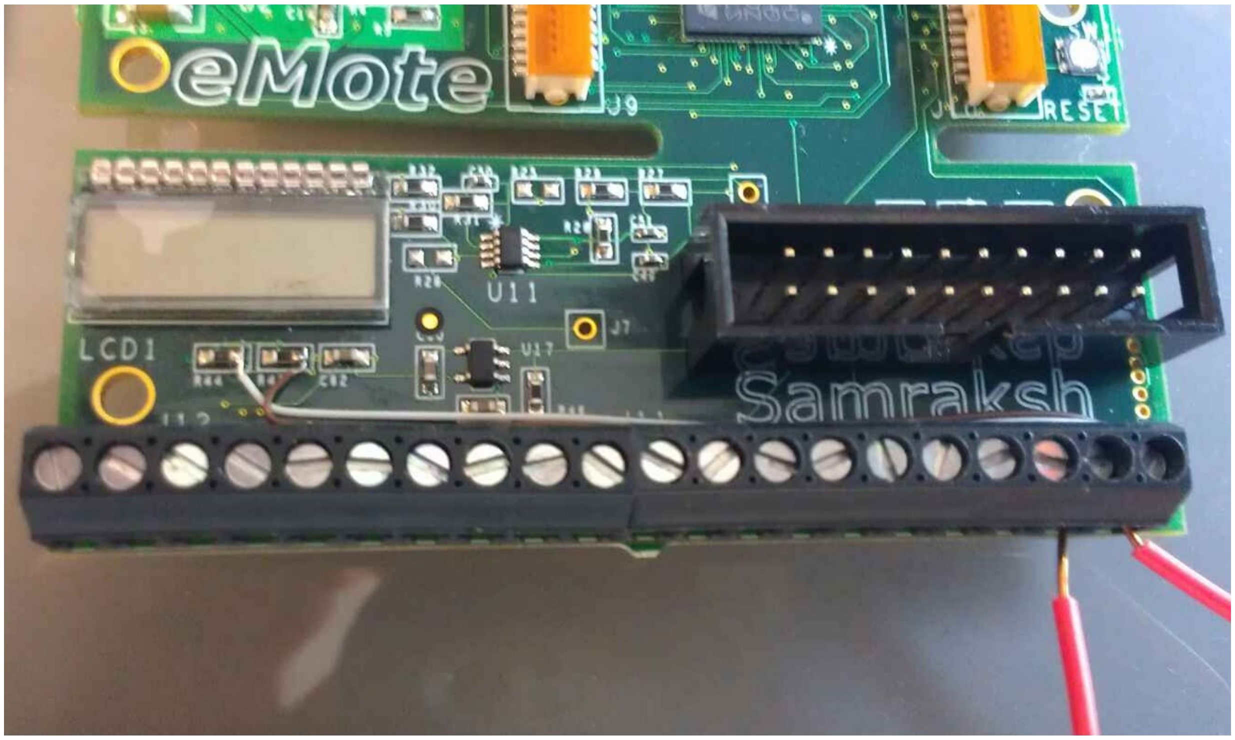
6. Appendix: Updating the eMote Firmware

From time to time, Samraksh will make new versions of the eMote software and firmware available. To use it, you will have to update eMote on your mote and then re-link your C# programs to use the new DLLs. In this section, we give you directions for updating eMote firmware.

First, make sure you've satisfied the prerequisites specified above in Section 3, Getting Started. You won't need Visual Studio for this.

Download the eMote .NOW firmware and DLLs. Go to samraksh.com and navigate to Support → eMote .NOW Downloads. Choose the version you want and click on it to download the ZIP archive. When downloaded, unzip it into a location of your choice.

Jumper pins J11/9 and J11/10 as shown by the colored wire below (that's connector J11 pin 9 and J11 pin 10; see .NOW components). Note that connector J11 sits directly to the right of J12, without a gap between them. With the jumper connected, the .NOW will boot into TinyBooter instead of TinyCLR.

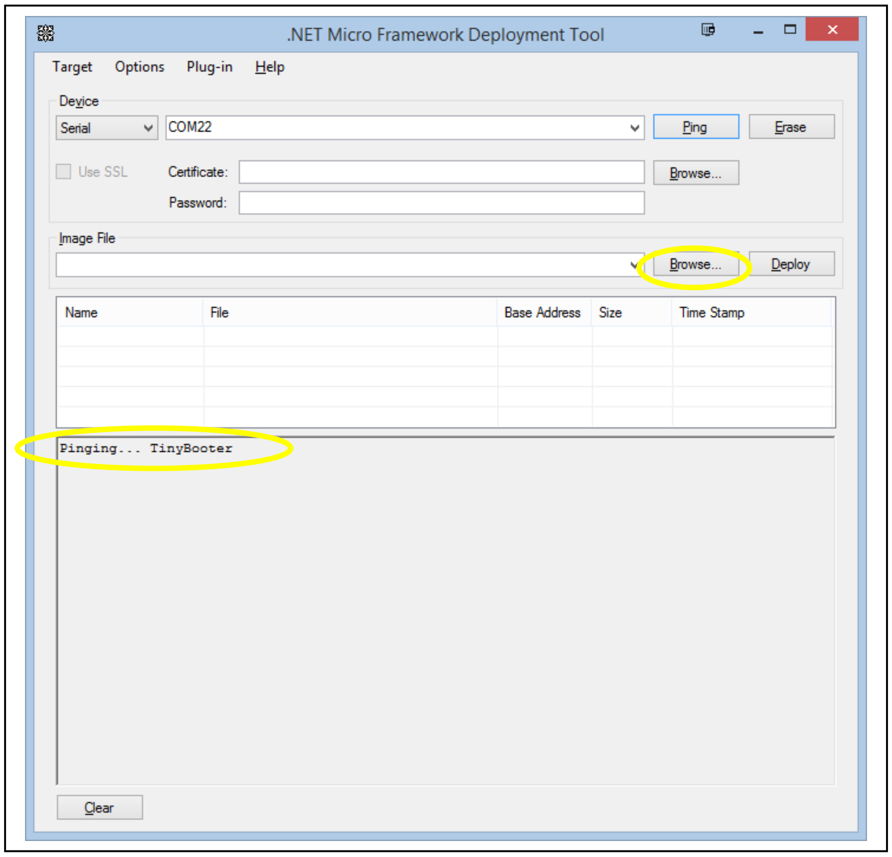


If there are any USB cables plugged into the .NOW board, remove them. This will power-down the board.

Plug a USB cable into the port shown, toward the top right of the .NOW board. Do not plug it into the port at the top of the board. Plug the other end into an available USB port on your PC. It might be best to use a USB port on the PC since USB port replicators sometimes do not work correctly.

Identify the COM port to use; see above in Section 3, Getting Started. Start MFDeploy and select the COM port; see above in Section 4.2:Using MFDeploy to Run and Test.

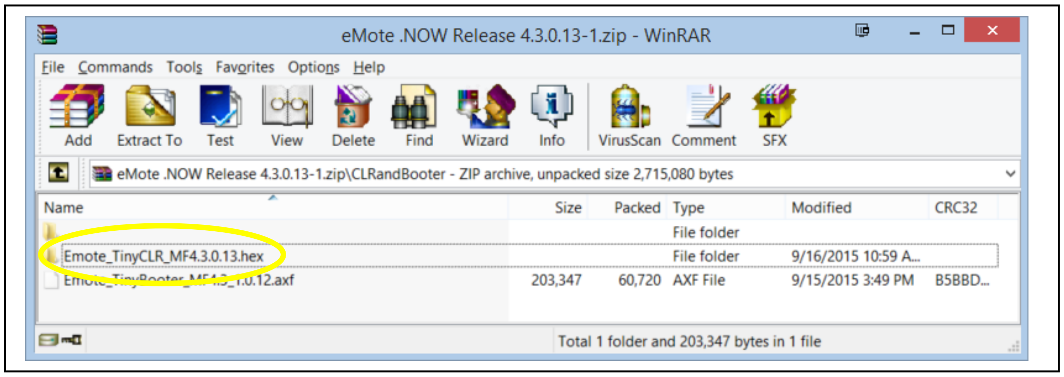
Click Ping. You should see TinyBooter.



Serial COM22

```
Pinging... TinyBooter
```

In MFDeploy, click on Browse under Image File. Navigate to the location where you unzipped the firm-ware and DLLs. Open the folder CLRandBooter, then open the folder with extension .hex.

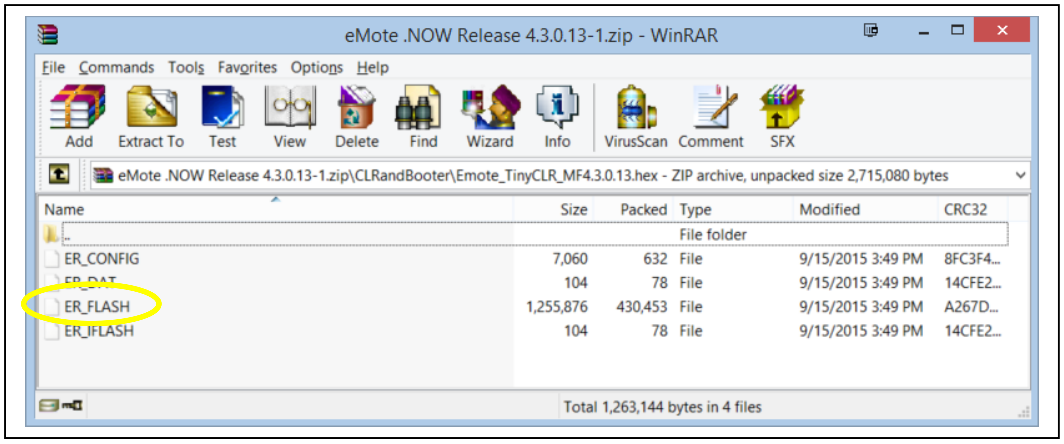


eMote .NOW Release 4.3.0.13-1.zip\CLRandBooter - ZIP archive, unpacked size 2,715,080 bytes

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
Emote_TinyCLR_MF4.3.0.13.hex			File folder	9/16/2015 10:59 A...	
Emote_TinyBooter_MF4.3_1.0.12.axf	203,347	60,720	AXF File	9/15/2015 3:49 PM	B5BBD...

Total 1 folder and 203,347 bytes in 1 file

Select the file named ER_FLASH. You may see other files present, but just ignore them. Click Open.

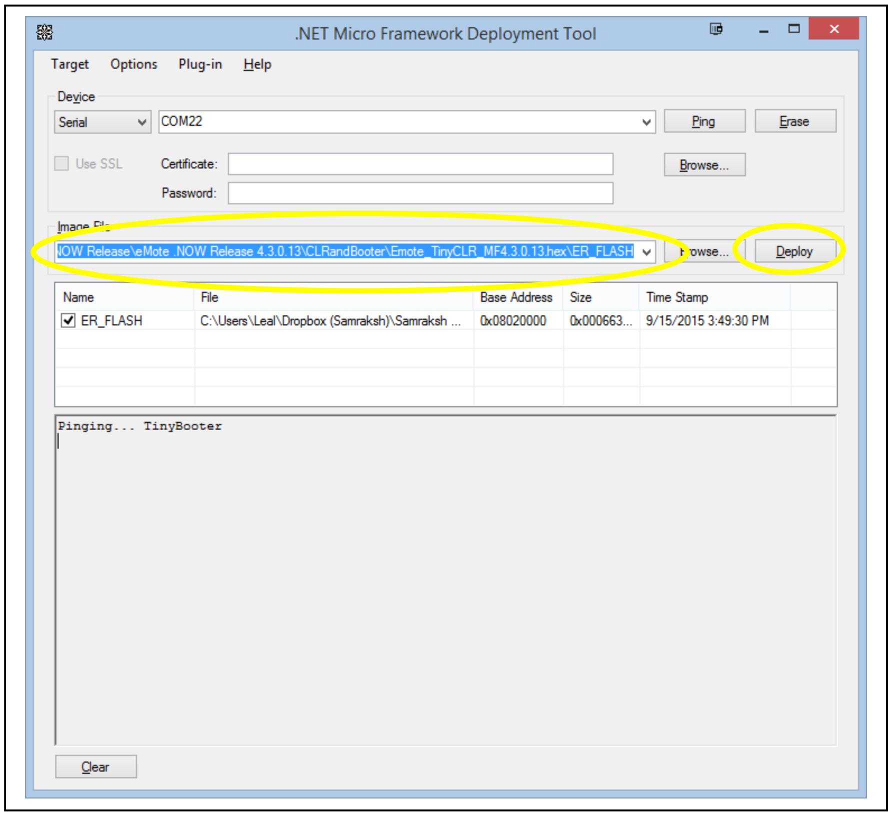

eMote .NOW Release 4.3.0.13-1.zip - WinRAR
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info VirusScan Comment SFX

eMote .NOW Release 4.3.0.13-1.zip\CLRandBooter\Emote_TinyCLR_MF4.3.0.13.hex - ZIP archive, unpacked size 2,715,080 bytes

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
ER_CONFIG	7,060	632	File	9/15/2015 3:49 PM	8FC3F4...
ER_DAT	104	78	File	9/15/2015 3:49 PM	14CFE2...
ER_FLASH	1,255,876	430,453	File	9/15/2015 3:49 PM	A267D...
ER_IFLASH	104	78	File	9/15/2015 3:49 PM	14CFE2...

Total 1,263,144 bytes in 4 files

In MFDeploy, you'll see the file listed below and you can click Deploy to flash them. You'll see messages in the log window as deployment progresses.


.NET Micro Framework Deployment Tool
Target Options Plug-in Help
Ping
Erase
Browse...
Deploy
Clear

Device: Serial COM22

Use SSL Certificate: Password:

Image File: NOW Release\eMote .NOW Release 4.3.0.13\CLRandBooter\Emote_TinyCLR_MF4.3.0.13.hex\ER_FLASH

Name	File	Base Address	Size	Time Stamp
☑ ER_FLASH	C:\Users\Leal\Dropbox (Samraksh)\Samraksh ...	0x08020000	0x000663...	9/15/2015 3:49:30 PM

```
Pinging... TinyBooter
```

Disconnect the jumper between J11/9 and J11/10. Unplug the USB cable to remove power. At this point you can plug the USB cable back in and begin using the eMote normally.